ภาษิต ภัทรมูล

ลีก ๆ กับ Visual FoxPro 5

เกร็ดลีก ๆ เกี่ยวกับ Visual FoxPro ทั้งทางด้านดีและด้านที่เป็นเงามืด

นที่สุด Visual FoxPro 5 ก็คลอดออกมา ท่ามกลางความสมหวังและผิดหวังคละเคล้ากันไป เรียกว่ามีทั้งข่าวดีและข่าวร้าย โปรแกรมเมอร์ตระกูล Fox ที่เพิ่งหายช็อกจากความงุ่มง่ามและโครงสร้างของ Class อันสลับซับซ้อนของ Visual FoxPro 3 โปรดประคองสติ แล้วฟังทางนี้

ข่าวดี Visual FoxPro 5 ได้ปรับปรุงความเร็วขึ้นเป็นที่น่าพอใจครับ

ข่าวไม่ค่อยดี โครงสร้างทางภาษาก็ยังคล้ายคลึงกับ Visual FoxPro 3

แต่นี่ก็ไม่ใช่เรื่องที่เราจะมาท้อถอย ยังมีทางแก้และทางเลี่ยงที่ให้เราเลือกทำ ในเมื่อลงทุนขยับขึ้นมาเขียนโปรแกรมบนวินโดวส์ก็ต้องทำใจและปรับตัวว่าระบบปฏิบัติการมันคนละแบบกับการเขียนบนดอสที่เราคุ้นเคยกันอยู่ เหมือนกับคำกล่าวที่ว่าการเปลี่ยนแปลงย่อมเป็นนิรันดร์

ใครที่เคยพัฒนาโปรแกรมบนดอสด้วย FoxPro, Clipper หรือ dBase พอหันมาเขียนบนวินโดวส์ก็ (จำใจ) ต้องเปลี่ยนไปใช้ภาษาที่ใช้ง่ายกว่า เช่น Access, Visual BASIC หรือ Delphi ซึ่งบางครั้งเขียน ๆ ไปก็ติดแหงกไปต่อไม่ได้ ด้วยความไม่คุ้นเคยกับไวยากรณ์ภาษาและข้อจำกัดของโปรแกรม เพราะฉะนั้น ถ้าจะลองหันมาชำเลือง Visual FoxPro 5 ก็ไม่ผิดกติกาอะไร อย่างน้อยความสันทัดจัดเจนที่บางคนเคยเป็น "เซียน FOX" ในดอสก็สามารถนำมาปรับใช้ในการพัฒนาโปรแกรมบนวินโดวส์ได้ ตรงนี้เป็นข้อได้เปรียบ เพราะการพัฒนาระบบข้อมูลที่ซับซ้อน การใช้ภาษาที่ฉนัดจะช่วยย่นระยะเวลาและมีบักน้อยลง

วัตถุประสงค์

ที่เขียนบทความนี้ขึ้นมา หวังว่าจะเป็นแนวทางสำหรับผู้ที่จะเขียนโปรแกรมบน Visual FoxPro ให้เป็นอาชีพและใช้งานได้จริง ดังนั้นจึงขอลงลึกในส่วนของปัญหาที่เราพบในการพัฒนาโปรแกรมบน Visual FoxPro ซึ่งทำให้หลายคนถึงกับต้องถอยมาตั้งหลัก แล้วถามตัวเองว่า เราจะมาบ้าแก้ปัญหาเล็ก ๆ น้อย ๆ อยู่ทำไม ในเมื่อใช้โปรแกรมตัวอื่นก็ไม่มีปัญหาแบบนี้

การแสดงผลเป็นภาษาไทย

พอเราเปิดเข้าไปในส่วนของหน้าต่าง Command Windows หรือในส่วนของหน้าต่างเขียนโค้ดของ Visual FoxPro ในคราวแรกมักจะพบว่า เมื่อเราพิมพ์ข้อความภาษาไทยลงไปแล้ว มันจะกลายเป็นภาษาที่อ่านไม่ออก ส่วนนี้ให้เปิดหาใน Menu Format เลือก Font เปลี่ยนฟอนต์ให้เป็นฟอนต์ภาษาไทยก็จะหมดปัญหา Menu Format-Font เราจะพบแทรกอยู่เกือบทุกจุดที่มีฟอนต์เข้ามาเกี่ยวข้อง

หลายคนคงกำลังปวดหัวกับการแสดงผลของฟอนต์ใน Screen ไม่รู้ว่าจะต้องไปเซตตรงไหน โดยเฉพาะเวลาใช้คำสั่ง @...say หรือ ? ที่ฉนัด ใน Visual FoxPro กลายเป็นตัวยึกยืออะไรไม่รู้

การทำให้ฟอนต์ใน Screen เป็นภาษาไทยมี 2 วิธี

1. ระบุฟอนต์ลงในประโยค @...Say,?,?? โดยตรง ตัวอย่าง

```
@10,20 say 'ทดสอบแสดงผลภาษาไทยด้วย IrisUPC'
font'IrisUPC',20
?'ทดสอบแสดงผลภาษาไทยด้วย IrisUPC' font'IrisUPC',20
```

วิธีนี้เราจะได้แสดงผลฟอนต์หลายชนิด หลากสไตล์ แต่ระวังตาลาย

2. ระบุลงไปในตัวแปร _Screen ขออธิบายเพิ่มเติมครับว่า Screen ของ Visual FoxPro ก็เป็น Object ตัวหนึ่ง ซึ่งก็มี Properties, Method, Events เช่นเดียวกับ Object อื่น ๆ เราสามารถอ้างถึงได้โดยผ่านตัวแปร _screen หรือ _vfp ตัวอย่าง

```
_screen.fontname='AngsanaUPC'
_screen.fontsize=20
```

วิธีนี้ เราจะได้ฟอนต์ชนิดและขนาดเดียวกันทั้ง Screen

หรือถ้าจะให้ดูเท่ากว่านั้น เราสามารถกำหนดให้ User เลือกฟอนต์มาแสดงด้วยตัวเองตามใจชอบ ดูตัวอย่างในลิสติงที่ 1 ครับ คงไม่ต้องอธิบายอะไรมากเพราะฟังก์ชันที่แปลกหน้าแปลกตา มีอยู่เพียงตัวเดียวคือ Getfont() ตัวนี้จะขึ้น Dialog Font ขึ้นมาให้เลือก ส่งค่ากลับเป็น Character

ลิสติงที่ 1

```
*selefont.prg
&& ให้ User เลือก Font มาแสดงใน Screen ด้วยตัวเอง
sfont=getfont()
      if Len(alltrim(sfont))<>0
      nfont=at(',',sfont)
      _screen.fontname=subs(sfont,1,nfont-1)
      _screen.fontsize=val(subs(sfont,nfont+1,2))
      endif
```

Table

Visual FoxPro ใช้ Table ในสองลักษณะคือ Database และ Free Table

- **Database** จะเก็บ Table หลาย ๆ Table ไว้ด้วยกันในแฟ้มมีนามสกุลว่า DBC พวกเราที่เคยใช้ฐานข้อมูลของ Access ซึ่งมีนามสกุลว่า MDB จะพบว่ามีส่วนคล้ายคลึง เพียงแต่ DBC ของ Visual FoxPro จะเก็บเฉพาะ Table เพียงอย่างเดียว การใช้ Database นี้มีทั้งข้อดีและข้อเสีย ข้อดีคือ เก็บหลาย ๆ Table ไว้ในที่เดียวกันสะดวกในการนำมาใช้และอ้างอิงถึง ส่วนข้อเสียคือ ไม่คอมแพทิเบิลกับ FoxPro เวอร์ชันก่อน ๆ กรณีที่เราเพิ่งเริ่มพัฒนาโปรแกรมใน Visual FoxPro ควรหลีกเลี่ยงไปก่อนจะดีกว่า

- **Free Table** ส่วนนี้จะเป็น Table เดี่ยว ๆ ยังคงแยกกันในลักษณะของแฟ้มนามสกุล DBF ซึ่งเราคุ้นเคยกันดี มีความคอมแพทิเบิลกับ FoxPro เวอร์ชันก่อน ๆ

ปัญหาที่เรามักพบกันมากคือ สร้าง Table ขึ้นมาแล้ว FoxPro ในดอสไม่รู้จัก ฟ้องว่า Not a table/DBF. (บางกรณีที่แฟ้มข้อมูลเกิดเสียหาย ใช้โปรแกรมบนดอสแก้จะสะดวกกว่า) นั่นเป็นเพราะว่า Visual FoxPro 5 ได้เพิ่มชนิดของ Field อีก 5 ชนิด คือ Currency, Datetime, Double, character (binary) และ Memo (binary) เราควรหลีกเลี่ยงการใช้ Field ประเภทนี้ นอกจากนั้นในไบต์ต้นแฟ้มจะมีโค้ดเป็น 0x30 ต่างจากต้นแฟ้มของ DBF ของ FoxPro For Dos จะเป็น 0x03 (No memo), หรือ 0xF5 (With memo)

ทางแก้เพื่อให้ใช้ฐานข้อมูลในดอสได้เหมือนเดิมก็คือ สร้าง Table ขึ้นโดย FoxPro For Dos แล้ว ADD เข้ามาใน Free Table ก็จะหมดปัญหาครับ

Code page เป็นอีกส่วนหนึ่งที่เราจะต้องปรับปรุง ทุกครั้งที่เรา Use แฟ้มข้อมูลที่สร้างมาจาก FoxPro for Dos จะขึ้น Prompt ขึ้นมาให้เราเลือก Code page ให้เลือก 874-Thai Windows/Ms-Dos แค่นี้ข้อมูลภาษาไทยในดอสของเราก็ยังจะใช้งานได้เหมือนเดิม

ถ้ารู้สึกรำคาญ Prompt ที่ขึ้นบ่อย ๆ เปิดเมนู tools เลือก Option เลือกหัวข้อ Data ปิดตรง Prompt for Code page เสีย

จากนั้นใช้เทคนิคการสร้าง Config ของเรามาให้ Visual FoxPro รู้จัก ให้ใช้ Text editor เช่น Notepad สร้างแฟ้มขึ้นมา ในที่นี้จะใช้ชื่อว่า Myconfig.fpw เก็บไว้ใน Directory c:\VFP พิมพ์ข้อความนี้ลงไป

```
CODEPAGE = 874
```

Save แฟ้มเก็บไว้ใช้ เมื่อเรารัน Visual FoxPro ในครั้งต่อไปให้ใช้คำสั่ง –C ตามหลัง VFP.EXE เช่น C:\vfp.exe –C c:\vfp\myconfig.fpw คราวนี้ Visual FoxPro ของเราก็จะไม่ขึ้น Prompt Code page มากวนใจ

Command Line ของ Visual FoxPro นี้มีประโยชน์มากครับ จะมีกล่าวถึงในช่วงต่อ ๆ ไป

เราสามารถเช็ค Code page ได้โดยใช้ function CPCURRENT()

การโปรแกรมและไวยากรณ์

ส่วนนี้ค่อนข้างจะหลากหลาย เนื่องจาก Visual FoxPro ได้เตรียมรองรับผู้ใช้ในหลายระดับและหลายสไตล์

น่าประทับใจในหน้าตาอันคุ้นเคยของ Command Windows ที่ติดตามเรามาตั้งแต่ในดอส ถ้าเราลองพิมพ์คำสั่งลงไป จะพบว่ามีการแสดงผลเหมือนในดอสทุกประการ มีประโยชน์สำหรับผู้ใช้ที่จะแก้ไขเปลี่ยนแปลงข้อมูลแต่ไม่คิดจะพัฒนาโปรแกรม จุดนี้แตกต่างจาก

โปรแกรมประเภท Visual ตัวอื่น ๆ นั่นคือ Visual FoxPro จะประกอบไปด้วย SCREEN (พื้นว่าง ๆ ที่เราเห็นเมื่อปิด Project Manager window) แทนที่จะโชว์ Form Designer และเจ้า Screen นี้จะติดตามเราไปตลอด รวมทั้งเมนูของ FoxPro ด้วย (ไม่ต้องตกใจนะครับ ถ้าผมจะบอกว่า แม้แต่ Project ที่เราคอมไพล์แล้ว มันก็ยังตามไปโชว์เหมือนเดิม แล้วจะบอกวิธีจัดการกับมัน) นอกจากนั้นก็ยังมีหน้าต่าง Browse, Edit, Change เช่นเดียวกับในดอส ส่วนที่แตกต่างจาก FoxPro For dos มาก ๆ คือ การใช้ Form ซึ่งเป็นส่วนสำคัญที่สุดก็ว่าได้ในการทำงานในระบบวินโดวส์ เราจะทุ่นเวลาในการสร้าง User interface ไปมากทีเดียว จากที่เคยเขียนโค้ดตกแต่งเป็นชั่วโมงอาจเหลือแค่ 15 นาทีเท่านั้น กรณีที่ใช้คล่องแล้วนะครับ มือใหม่คงนานกว่านั้น

การสร้างฟอร์มสามารถสร้างได้ 2 ลักษณะ

1. การเขียนโค้ด ตรงนี้แหละครับที่เราต้องไปยุ่งกับการใช้ Class ดูแล้วน่าสับสนไม่น้อย เข้าใจว่าทีมผู้พัฒนา Visual FoxPro ต้องการจะเอาใจเหล่าโปรแกรมเมอร์ที่ใช้ OOP (Object-Oriented Programming) เพราะไม่ค่อยจะเข้ากับผู้ใช้ FoxPro โดยทั่วไป แถมยังทำให้ดูยุ่งยากขึ้นโดยใช่เหตุ จะขอนำเสนอพอเป็นสังเขป เนื่องจากเราหลีกเลี่ยงไม่ใช้ Class เลยก็ได้

โครงสร้างหลักของ Class

ขึ้นต้นด้วย Define class <ชื่อ Class> as <ชนิดของ object>

...กำหนดคุณสมบัติของ Class...

...โปรแกรมย่อย/ฟังก์ชัน ที่เกี่ยวข้องกับ Events ของ Class (มีประมาณ 32 ชนิด) และโปรแกรมย่อย/ฟังก์ชันอื่น ๆ ที่ต้องใช้ใน Class หรือเราจะเรียกโปรแกรมย่อย/ฟังก์ชันจากภายนอกก็ได้ โดยใช้คำสั่ง Set Procedure to <ชื่อแฟ้มที่เก็บโปรแกรมย่อย/ฟังก์ชัน>

จบลงด้วย Enddefine

ตัวอย่างเช่น เราจะสร้าง Form ขึ้นมาสักตัว ชื่อว่า MYFORM มี Title ว่า Hello! ใน Form นั้นมี Command button ตัวหนึ่ง ชื่อว่า MYbutt ให้มี Caption ว่า "OK" เมื่อคลิกเมาส์ลงไปที่ Command Button ให้ขึ้นเมสเสจบ็อกซ์ ว่า 'คุณกด OK'

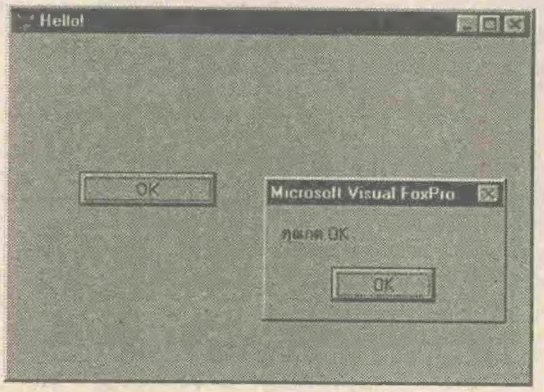
ก่อนอื่น เราต้องสร้าง Form กำหนดคุณสมบัติ บอกชนิดของ Object ที่มีใน Form (ดูลิสติงที่ 2)

ความจริงเราสามารถสร้าง Class ของ Form ขึ้นมาด้วยก็ได้ แต่ในที่นี้จะเน้นการทำ Class ของ Command button ขึ้นมาใน MyForm เมื่อพิจารณาแล้วจะเห็นว่าโครงสร้างของ Class ก็ไม่มีอะไรซับซ้อน แต่ที่หนักใจก็ตรงเขียนโค้ดยาวนี่แหละ! พิมพ์ตาม แล้วลองรันได้เลยครับ

ลิสติงที่ 2

```
Myform=CREATEOBJECT('Form')          &&สร้าง Form
Myform.caption='Hello!'              &&กำหนด Title ของฟอร์ม
Myform.AddObject("mybutt","mybuttclss") &&เพิ่ม Object ในฟอ
ชื่อ mybutt ที่มาจากคลาส Class "Mybuttclss"
Myform.mybutt.visible=.t.            &&ให้ Mybutt ปรากฏใน Form
Myform.show                          &&แสดง Form ขึ้น
Read events                          &&เริ่มตรวจจับ Event ที่ปรากฏบน Form
*สร้าง Class ต่อท้าย ส่วนที่เรียกใช้
DEFINE CLASS MYBUTTCLSS as commandbutton
Caption='OK'                         &&กำหนด Caption
Left = 50                            && ตำแหน่งแนวนอน
Top = 100                            && ตำแหน่งแนวตั้ง
Height = 25                          && ความสูง

PROCEDURE Click
    =Messagebox('คุณกด OK')
ENDDEFINE
```



รูปที่ 1

จะเห็นว่าการแสดงผลของ Form ที่ถูกสร้างด้วยโค้ดนั้น ทำงานได้รวดเร็วน่าประทับใจมาก (ดูรูปที่ 1)

ถ้าคุณอยากเข้าใจเรื่อง Class ให้มากขึ้น ให้ลองพิจารณาซอร์สโค้ดตามลิสติงที่ 3 จะลองพิมพ์ตามแล้วรันดูจะกระจ่างขึ้น

ลิสติงที่ 3

```
MyForm = CREATEOBJECT('Form')
myform.Closable = .F.
myform.height=200
myform.autocenter=.t.
myform.caption='Create form by code!'

myform.AddObject('boxx','shape')
myform.AddObject('cmd1','MyCmndBtn1')
myform.AddObject('cmd2','MyCmndBtn2')
myform.AddObject('cmd3','MyCmndBtn3')

myform.boxx.Visible = .T.
myform.boxx.Top = 20
myform.boxx.height=70
myform.boxx.Left = 125

myform.cmd1.Visible =.T.
myform.cmd2.Visible =.T.
myform.cmd3.Visible =.T.

myform.SHOW
READ EVENTS
```

(มีต่อ) →

ลิสติงที่ 3 (ต่อ)

```
DEFINE CLASS MyCmndBtn1 AS COMMANDBUTTON
      Caption = 'น้ำเงิน'
      Left = 50
      Top = 100
      Height = 30
      Fontname='IrisUPC'
      Fontsize=18

      PROCEDURE Click
            ThisForm.boxx.backcolor=rgb(0,0,225)
ENDDEFINE

DEFINE CLASS MyCmndBtn2 AS CommandButton
      Caption = 'แดง'
      Left = 200
      Top = 100
      Fontname='IrisUPC'
      Fontsize=18

      PROCEDURE Click
            ThisForm.boxx.backcolor =rgb(255,0,0)
ENDDEFINE

DEFINE CLASS MyCmndBtn3 AS CommandButton
      Caption = 'ออก'
      Cancel = .T.
      Left = 250
      Top = 150
      Fontname='IrisUPC'
      Fontsize=18

      PROCEDURE Click
            CLEAR EVENTS
ENDDEFINE
```

เรื่องราวเกี่ยวกับ Class ยังมีอีกมาก ใครที่ชอบอะไรยาก ๆ โหดและเมื่อย (มือ) แต่ประสิทธิภาพดี เชิญค้นคว้าต่อได้เลยครับ

2. การสร้าง Form โดยใช้ Form designer (ซึ่งคนธรรมดาส่วนใหญ่เขาทำกัน) ผมคงจะไม่กล่าวถึงการสร้าง เพราะเนื้อหามีมากเกินไปสำหรับบทความนี้ ตำรับตำราที่กล่าวถึงเรื่องนี้ก็พอมีจำหน่ายตามท้องตลาด แต่ครั้งนี้จะกล่าวถึงปัญหาที่เรามักพบในการเรียกใช้ Form ดังนี้

- เมื่อใช้ Form ในลักษณะ FORM หลัก/Form ย่อย Screen มักจะขึ้นมาแทรกกวนใจ หรือหลังจากปิด Form ย่อยแล้ว Form หลักไม่ยอมขึ้นโชว์ โปรแกรมหลุดมาที่ Screen เลย ถ้าใครอ่านตรงนี้แล้ว งง ๆ ให้ลองเรียกใช้ Form ในลักษณะนี้ดูได้เลย Visual FoxPro มีแนวคิดต่างจากโปรแกรมแบบ Visual อื่น ๆ อย่างเห็นได้ชัด ส่งผลให้ Form ที่สร้างเมื่อเรียกใช้เกิดผลลบ ๆ โผล่ ๆ อย่างไรชอบกล ไม่เชื่อก็ต้องเชื่อครับ มีโปรแกรมเมอร์คนหนึ่งถึงขนาดตัดญาติขาดมิตรกับ Visual FoxPro ไปเลย ด้วยเหตุผลที่ว่า ควบคุม Form ไม่ได้ดังใจ!

ความจริงการควบคุม Form ในลักษณะนี้ มีอยู่ใน Properties ตัวเดียวแท้ ๆ นั่นคือ SHOW Windows ให้คุณตั้ง Form หลักให้เป็น 2-as top-level form ส่วน Form ย่อยที่ Form หลักเรียกใช้ตั้งเป็น 1-in top-level form รับรองว่า Form ของคุณจะอยู่ในระเบียบเป็นอันดี ไม่เกะกะเกเรอีกต่อไป

- Screen และ Menu ยังมีอยู่หลังจากคอมไพล์โปรแกรมแล้ว แถมยังโชว์ Title ว่า Microsoft Visual FoxPro อีกต่างหาก เห็นแล้วบาดใจจริง ๆ

ทางแก้มีวิธีดังนี้

1. เอาแฟ้ม Myconfig.fpw ที่เราสร้างขึ้นในตอนต้น ๆ บทความมาแก้โดยเพิ่มบรรทัดว่า

```
Title=<ประโยคข้อความ>
เช่น Title=จ๊าบ 1.04
เมื่อเรียกใช้โปรแกรมก็ใช้ Command line -C เหมือนที่เคยกล่าวถึงมาแล้ว
```

2. ใช้โค้ดในช่วงต้น ๆ ของโปรแกรมดังนี้ (ลิสติงที่ 4)

ลิสติงที่ 4

```
_Screen.caption='จ๊าบ 1.04'      &&เปลี่ยนประโยคใน Title
set sysmenu off                  &&ทำลายเมนู
```

ส่วนประกอบของ Project

พวกเราที่เคยชินกับการเขียนโปรแกรมในสไตล์ดอส เคยเขียนแฟ้ม .Prg ตัวเดียวแล้วเรียบร้อยด้วย Procedure และ Function มีการใช้ Table และ Report โดยการพิมพ์คำสั่งอย่างเดียว คงต้องเปลี่ยนแล้วครับ เพราะ Visual FoxPro สนับสนุนการทำงานในลักษณะเป็น Project คือ ใน Project จะเก็บสิ่งสารพัดสารพันในการทำงานของโปรแกรมที่เราสร้างไว้ในนั้น ตอนคอมไพล์มันจะเลือกคอมไพล์เฉพาะไฟล์ที่มีอยู่ใน Project นั้น ๆ ไม่ต้องไปล้วงหาเอาเองในฮาร์ดดิสก์ ถ้ายังรักจะเขียนโปรแกรมด้วยคำสั่งอย่างเดียวเหมือนเดิม ขอให้กลับไปดูเรื่อง Class ใหม่ คงจะเป็นหนทางที่โหดร้ายกับตัวเองเกินไปนะครับ

ที่เราควรรู้

เรามาดูว่า มีอะไรที่เรายังคงทำได้ และอะไรที่ต้องเปลี่ยนแปลงหรือต้องมาเรียนรู้เพิ่มเติม

Procedure หรือ Function

การเรียกใช้/ส่ง/รับค่าตัวแปรยังเหมือนเดิม แต่ที่ต้องรู้คือ Procedure ที่ถูกกำหนดให้เป็น Events เช่น Procedure click เป็นต้น (ส่วนใหญ่เราจะใช้มากในการทำงานกับ Form) ไม่สามารถเขียน Function หรือ Procedure อื่นอยู่ในตัวมันได้ การเรียกใช้ Procedure หรือ Function ข้างนอกให้ใช้ Set Procedure to ช่วย

การสั่งให้ Project ทำงานหลังจากคอมไพล์แล้ว

Project จะทำงานโดยรัน Form หรือ Program ที่ถูกเซตให้เป็น Main

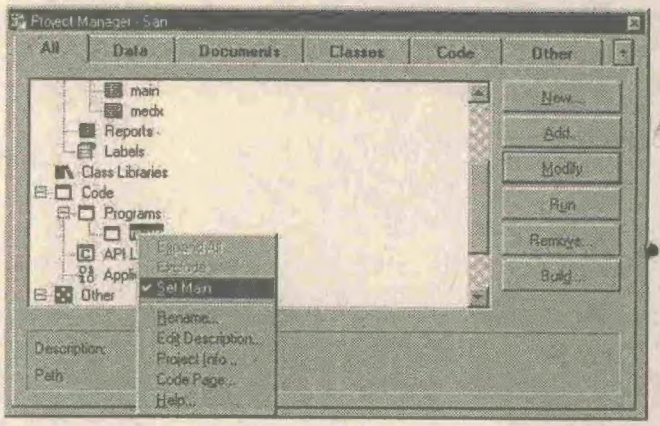
โครงสร้างหลักของ Main.prg
-ส่วนการกำหนดค่า และตัวแปรต่าง ๆ
-เรียกใช้เมนูหลัก
-เรียกใช้ Screen หรือฟอร์มหลัก
-ลบล้างการกำหนดค่า/ตัวแปร
-ออก

ตัวอย่างของโปรแกรมที่เป็น Main ดูลิสติงที่ 5

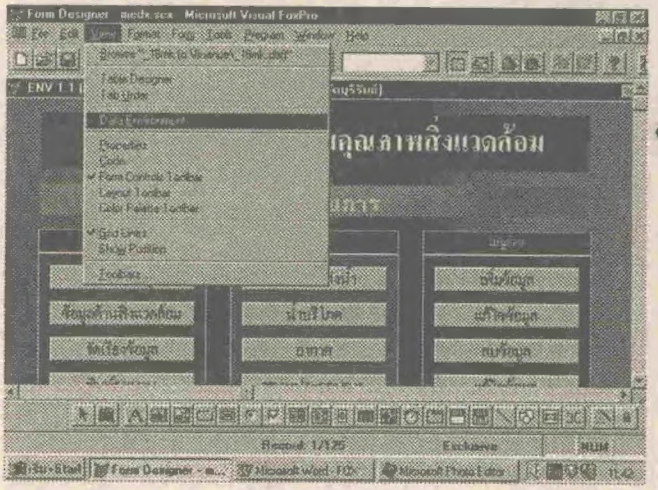
ลิสติงที่ 5

```
***main.prg
&&กำหนดค่าและตัวแปรระบบ
set talk off
set date british
set century on
set procedure to mainproc
&&เรียกใช้ Menu และ Form หลัก
do menu.mpr
do form formmain
read events
&&ลบตัวแปรต่าง ๆ ออกจากโปรแกรม
clear events
```

การ Set Main ให้คลิกไปที่โปรแกรมที่ต้องการให้เป็น Main แล้วคลิกปุ่มขวาของเมาส์ จะมีเมนูขึ้นมาให้เลือก (ดูรูปที่ 2) เลือก set main จะมีผลเมื่อตอนคอมไพล์แล้ว Program ที่เป็น Main จะถูกเรียกใช้ก่อน



รูปที่ 2



รูปที่ 3

สิ่งที่เปลี่ยนแปลงใน Form, Report, label

จะมี Data environment เพิ่มขึ้นมา หากเราจะนำเอาค่าของ Field ใน Table ใดขึ้นมาแสดง Table นั้นจะต้องปรากฏอยู่ใน Data environment มิฉะนั้นโปรแกรมจะฟ้องขึ้นว่า หา Table นั้น ๆ ไม่เห็น แม้ว่าจะ Add มาไว้ใน Project แล้วก็ตาม ถ้าเราใช้ Form designer คลิกที่เมนู View เลือก Data Environment (ดูรูปที่ 3) เราสามารถเข้าไปแก้ไขได้เลย (ใน Report Designer ก็เช่นเดียวกัน)

การอ้างถึงค่า Field ใน Table ที่ปรากฏใน Data environment แล้วให้พิมพ์ชื่อ Table ตามด้วยจุด (.) แล้วชื่อ field ตัวอย่าง Customer.name สามารถใช้ร่วมกับคำสั่งของ FoxPro เช่น Replace, store to var, substring, trim, alltrim ฯลฯ ได้ตามปกติ

ไดอะล็อก Box เท่ ๆ

ขออนุญาตไม่แสดงตัวอย่าง แต่อยากให้พวกเราลองพิมพ์ตามข้างล่างที่ Command Windows แล้วจะเห็นเอง

```
?getfile()
?getcolor()
?getfont()
?getcp()
?getdir()
?getpict()
?getprinter()
```

Messagebox เท่ ๆ

ป้อนโค้ดตามลิสติงที่ 6 แล้วรันได้เลย (รับรองว่าจะประทับใจในความง่ายในการจัดการกับ Message ของ Visual Foxpro)

ลิสติงที่ 6

```
*Mes_ex.prg
&&ตัวอย่างการระบุเงื่อนไขให้กับข่าวสารที่แจ้งให้ผู้ใช้ตัดสินใจ
set path to (home())
#include FOXPRO.H
ox=Messagebox("คุณรู้สึกหิวหรือ
ไม่",MB_ICONQUESTION+MB_YESNO)
if ox=IDYES
wait window "คุณควรจะรับประทานอาหารเสียตอนนี้" timeout 5
else
wait window "ถ้าอย่างนั้น ทำงานต่อได้เลย!" timeout 5
endif
```

ถ้าต้องการจะปรับการแสดงผลและไอคอนประกอบที่แตกต่างจากนี้ ให้เปิดดูในแฟ้ม FoxPro.h ได้ครับ

Menu เจ้าปัญหา

ใช้ Wizard สร้างเมนูพอรันเมนูทีไรเป็นต้องเกิด Error ทุกครั้ง ทางป้องกันและแก้ไขคือ ต้องไปกำหนด Shortcut ให้เมนูด้วยทุกครั้ง หรือ generate แล้วตามไปลบบรรทัดที่มีคำว่า key ทิ้งไป

USE AGAIN?

ใน Visual FoxPro 5 เราสามารถ Use แฟ้มข้อมูลเดียวกันในต่างพื้นที่ได้ แต่ต้องมีประโยค Again ต่อท้าย เช่น

```
USE customer in 1
USE customer in 2 again
```

Object ActiveX

ActiveX ที่ Microsoft คุยนักคุยหนาว่าสามารถใช้ร่วมกับ Visual FoxPro ได้เป็นอย่างดี ตอนนี้ เท่าที่รู้มีหลายตัวที่ใช้ไม่ได้ แม้แต่ Progress bar ยังไม่มีปัญญาใช้เลยครับ ผมจึงต้องสร้าง Progress bar ขึ้นมาใช้เอง ตามมีตามเกิด เพราะการประมวลผลบางอย่างต้องใช้เวลานาน จะให้เครื่องเงียบไปเฉย ๆ โดยไม่มีข่าวสารแจ้งแก่ผู้ใช้ก็ดูกระไรอยู่ ถ้ามีโอกาสจะเขียนเกี่ยวกับการสร้าง Progress Bar ให้อ่าน

การใช้ Visual FoxPro 5 กับ HTML

ตรงนี้ขอติดไว้ครับ เพราะยังไม่ได้ลองใช้

SETUP DISK

การทำแผ่นติดตั้งโปรแกรม หลังจากพัฒนาระบบเสร็จเรียบร้อยแล้ว เนื้อหามีมากพอสมควร ขอผลัดเป็นครั้งหน้า ส่วนนี้มีความสำคัญไม่น้อย เพราะพัฒนาโปรแกรมเสร็จแล้ว แต่นำไปลงเครื่องอื่นไม่ได้ มันก็เจ๊งลูกเดียว...

ของแถม

ตรงนี้เกี่ยวข้องกับความสวยงาม โปรแกรมถ้าขาดซึ่งสีสัน คงจะดูจืดชืดไม่น่าใช้ บางคนอาจไปเห็นภาพสวย ๆ แต่เป็นแฟ้มสกุล jpg, GIF, PCX หรืออื่น ๆ ที่ Image Controls ของ Visual FoxPro ไม่รู้จัก อยากจะดึงมาประดับที่ Form ถ้าต้องเสียเวลาเอาไปแปลงฟอร์แมตเป็นสกุล BMP ดูจะไม่ค่อยคุ้มค่า ขอเสนอเทคนิคเล็กน้อย สำหรับเครื่องที่มี Ms.office 97 ติดตั้งอยู่ในเครื่องแล้วเท่านั้นนะครับ

ที่ Form Designer เลือก OLE container control แล้วนำไปวางที่ Form จะมีวินโดว์ขึ้นมาให้เลือก Microsoft Clipart Gallery แล้ว Add Picture ที่เราต้องการมาไว้ในแคตตาล็อก จากนั้น Insert เอามาใช้ได้เลย เทคนิคนี้ยังใช้ได้กับ Delphi, MS-access ทุกเวอร์ชัน!

ส่งท้าย

เกร็ดลีก ๆ เกี่ยวกับ Visual FoxPro ทั้งทางด้านดีและด้านที่เป็นเงามืด อาจสรุปให้เราเห็นภาพกว้าง ๆ ว่า การกลับมาของ Visual FoxPro 5 ในครั้งนี้ มีหลายอย่างที่ปรับปรุงขึ้นมาเป็นที่น่าพอใจ โดยเฉพาะในด้านความเร็วที่ FoxPro เคยเป็นจ้าวด้านความเร็วในโปรแกรมประเภทฐานข้อมูลด้วยกันมาแต่เดิมและต้องสูญเสียตำแหน่งไปเมื่อเป็น Visual FoxPro 3 บัดนี้ก็กลับมาทวงตำแหน่งกลับคืน ต้องเดาใจแฟน ๆ ที่ติดตาม FoxPro มาแต่เดิมล่ะครับ ว่าจะกลับมาย้อนความหลังกับโปรแกรมตัวเก่าในร่างใหม่ ซึ่งมีการเจริญเติบโตและเปลี่ยนแปลงตามกาลสมัย หรือว่าจะลืมเลือนมันไป จะอภัยให้เรียมหรือเปล่า

เนื้อที่นำเสนอไปข้างต้น อาจจะยังไม่สมบูรณ์นัก เพราะปัญหาที่นักพัฒนาแต่ละคนได้พบอาจแตกต่างกันไป เป็นเฉพาะตัวบุคคล รวมถึงฮาร์ดแวร์และซอฟต์แวร์ล้วนเป็นสภาพแวดล้อมที่มีผลต่อการพัฒนาทั้งสิ้น อย่างไรก็ตาม ผู้เขียนได้พยายามนำเสนอเฉพาะในประเด็นที่พบบ่อย และต้องใช้ความรู้ด้านเทคนิคมาใช้ในการแก้ไข ข้อมูลเหล่านี้ที่ไม่ค่อยอ่านเจอใน Help หรือหนังสือตำรับตำราทั่วไป จึงน่าจะเป็นประโยชน์ต่อผู้สนใจตามสมควร ยินดีรับทราบปัญหาและคำแนะนำเพิ่มเติมของผู้อ่านทุกท่านครับ